**IN THE CLAIMS:**

1. (currently amended) An object oriented ~~computing~~ computer system on [[a]] at least one computer platform, comprising:

objects comprising software components which are dynamically loadable by a user at user runtime into said at least one computer platform and which have dynamically linkable named inputs and outputs named by the user at runtime and also modifiable by the user at runtime stored on a memory of the computer system so that at runtime the user can define or modify a functionality of a configuration of the software components, said components also having internal tasks for queuing of data transferred into and out from the components via said inputs and outputs; and

an event communication framework providing automated, pattern-based, fully distributable events such that when a new dynamically loadable at user runtime software component is loaded into said computer system also having dynamically linkable named inputs and outputs, the new software component inputs and outputs are all automatically linked to the inputs and outputs of the same name of said stored software components, so that the new and stored software components are combined without changing any code within the software components and without writing any adapters.

2. (currently amended) The object oriented ~~computing~~ computer system of claim 1, wherein the inputs and outputs of the objects are provided via CsaConnectable and CsaRemote objects, respectively.

3. (currently amended) The object oriented ~~computing~~ computer system of claim 2, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.

4. (currently amended) The object oriented ~~computing~~ computer system of claim 2, wherein each object is a shared library which is dynamically loadable at runtime by an ASCII configuration file containing the names of the named inputs and outputs of the objects.

5. (currently amended) An object oriented ~~computing~~ computer system on [[a]] at least one ~~computing~~ computer system, comprising:

a memory of the ~~computing~~ computer system storing objects;

said objects comprising software components which are dynamically loadable by a user at user runtime into said at least one computer system and having dynamically linkable ~~named~~ inputs and outputs named by the user at runtime and also modifiable by the user at runtime so that at runtime the user can define or modify a functionality of a configuration of the software components, and internal tasks for queuing of data transferred into and out from the objects via said inputs and outputs, respectively; and

an event communication framework providing automated, pattern-based, fully distributable events such that when a new software component is loaded at user runtime into said computer system also having dynamically linkable named inputs and outputs, the new software component inputs and outputs are all automatically linked to the inputs and outputs of the same name of said stored software components so that the new and stored software components are combined without changing any code and without writing any adapters.

6. (currently amended) The object oriented ~~computing~~ computer system of claim 5, wherein the inputs and outputs of the objects are provided via CsaConnectable and CsaRemote objects, respectively.

7. (currently amended) The object oriented ~~computing~~ computer system of claim 6, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.

8. (currently amended) The object oriented ~~computing~~ computer system of claim 6, wherein each object is a shared library which is dynamically loadable at runtime by an ASCII configuration file containing the names of the named inputs and outputs of the objects.

9. (currently amended) A method for designing software components in an object oriented ~~computing~~ computer, comprising the steps of:

defining input and output events that are fully distributable;

configuring software components which are dynamically loadable by a user at user runtime by dynamically linkable ~~named~~ input and output connection points named by the user at runtime and also modifiable by the user at runtime and storing the components on a memory of the computer system so that at runtime the user can define or modify a functionality of a configuration of the software components, said components also having internal tasks for queuing of data transferred into and out from the components via said input and output connection points; and

providing autorouted pattern based fully distributable events based on an event communication framework such that when a new dynamically loadable at user runtime software component is loaded into said computer system also having dynamically linkable named input and output connection points, the new software component input and output connection points are all automatically linked to the input and output connection points of the same name of said stored software components, so that the software components are combined without changing any code within the software components and without writing any adapters.

10. (currently amended) A storage medium including object oriented code having an object oriented ~~computing~~ computer system on a computer platform, comprising:

objects comprising software components which are dynamically loadable by a user at user runtime into said at least one computer platform and having dynamically linkable named inputs and outputs named by the user at runtime and also modifiable by the user at runtime stored in memory of the computer system so that at runtime the user can define or modify a functionality of a configuration of the software components, said components also having internal tasks for queuing of data transferred into and out from the components via said inputs and outputs; and

an event communication framework providing automated, pattern-based, fully distributable events such that when a new dynamically loadable at user runtime software component is loaded into said computer system also having dynamically linkable named inputs and outputs, the new software component inputs and outputs are all automatically linked to the inputs and outputs of the same name of said stored software components, so that the software components are combined without changing any code within the software components and without writing any adapters.

11. (original) The storage medium of claim 10, wherein the inputs and outputs of the objects are provided via CsaConnectable and CsaRemote objects, respectively.

12. (original) The storage medium of claim 11, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.

13. (previously presented) The storage medium of claim 12, wherein each object is a shared library which is dynamically loadable at runtime by an ASCII

configuration file containing the names of the named inputs and outputs of the objects.

14. (currently amended) A storage medium, comprising:

object oriented code for an object oriented ~~computing~~ computer system on a ~~computing~~ computer system;

objects comprising software components which are dynamically loadable by a user at user runtime into said computer system and stored on a memory of the computer system and having dynamically linkable named inputs and outputs named by the user of runtime and also modifiable by the user at runtime so that at runtime the user can define or modify a functionality of a configuration of the software components and internal tasks for queuing of data transferred into and out from the objects via said inputs and outputs, respectively; and

an event communication framework providing automated, pattern-based, fully distributable events such that when a new software component at user runtime is loaded into said computer system also having dynamically linkable named inputs and outputs, the new software component inputs and outputs are all automatically linked to the inputs and outputs of the same name of said stored software components, so that the software components are combined without changing any code of the software components and without writing any adapters.

15. (original) The storage medium of claim 14, wherein the inputs and outputs of the objects are provided via CasConnectable and CsaRemote objects, respectively.

16. (original) The storage medium of claim 15, wherein each data structure associated with the inputs and outputs is described in a separate header file which can be used by every object to be linked.

17. (previously presented) The storage medium of claim 15, wherein each object is a shared library which is dynamically loadable at runtime by an ASCII configuration file containing the names of the named inputs and outputs of the objects.

18. (currently amended) A method for designing software components in an object oriented ~~computing~~ computer system having a storage medium including object oriented code, comprising the steps of:

defining input and output events that are fully distributable;

configuring software components which are dynamically loadable by a user at user runtime into said computer system by dynamically linkable named input and output connection points named by the user at runtime so that at runtime the user can define or modify a functionality of a configuration of the software components and stored on a memory of the computer system so that at runtime the user can define or modify a functionality of a configuration of the software components, said components also having internal tasks for queuing of data transferred into and out from the components via said input and output connection points; and

providing autorouted pattern based fully distributable events based on an event communication framework such that when a new software component is loaded at user runtime into said computer system also having dynamically linkable named input and output connection points, the new software component input and output connection points are all automatically linked to the input and output connection points of the same name of said stored software components, so that the software components are combined without changing any code within the software components and without writing any adapters.